I Erläuterungen

Voraussetzungen gemäß KCBG und Abiturerlassen BG jeweils in der für den Abiturjahrgang geltenden Fassung

Standardbezug
Die nachfolgend ausgewiesenen Kompetenzbereiche sind für die Bearbeitung der jeweiligen Aufgabe besonders bedeutsam. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Kompetenzen für die Bearbeitung der Aufgabe nachrangig bedeutsam sein, zumal die Kompetenzen in engem Bezug zueinander stehen. Die Operationalisierung des Bezugs zu den Kompetenzbereichen des Standardbezugs erfolgt in Abschnitt II.

Aufgabe	Kompetenzen				
	K1	K2	K3	K4	K5
1.1	X	X			
1.2				X	
1.3		X	X		
1.4				X	
1.5				X	
1.6			X		
1.7.1	X	X			
1.7.2		X	X		
2.1				X	X
2.2	X	X			
2.3.1				X	
2.3.2				X	
2.3.3			X	X	
2.3.4			X	X	
2.3.5			X	X	
3		X	X		

Inhaltlicher Bezug
Die nachfolgend ausgewiesenen Themenfelder sind die wesentliche inhaltliche Grundlage für die vorliegenden Aufgaben. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Themenfelder für die Bearbeitung nachrangig bedeutsam sein.
Q1: Objektorientierte Softwareentwicklung
Q2: Datenbanksysteme
verbindliche Themenfelder: Objektorientierte Modellierung (Q1.1), Implementierung von Klassen und Assoziationen (Q1.2), Datenstrukturen (Q1.4), Konzeptionelle und logische Modellierung einer Datenbank (Q2.1), Datenabfrage und Datenmanipulation mit SQL (Q2.2)

II Lösungshinweise

In den nachfolgenden Lösungshinweisen sind alle wesentlichen Gesichtspunkte, die bei der Bearbeitung der einzelnen Aufgaben zu berücksichtigen sind, konkret genannt und diejenigen Lösungswege aufgezeigt, welche die Prüflinge erfahrungsgemäß einschlagen werden. Selbstverständlich sind jedoch Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, ebenso zu akzeptieren.

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.1	beschreiben Die abstrakte Klasse `Komponente` als Oberklasse (Superklasse) dieser Vererbungsstruktur ist eine Generalisierung der Unterklassen (Subklassen) `RahmenTeil`, `SchaltElement` und `EinzelTeil`. Die Subklassen haben die gemeinsamen Attribute `iD`, `bezeichnung`, `gewicht`, `preis` und `hersteller`, die in die Superklasse ausgelagert sind. Außer der privaten Identifikationsnummer haben diese Attribute das Zugriffsrecht `protected` (geschützt), benötigen innerhalb der Vererbungshierarchie beziehungsweise aus Klassen des Pakets keine get- und set-Methoden, sind aber gegen Zugriffe von außerhalb geschützt. Die drei Subklassen sind Spezialisierungen ihrer Superklasse `Komponente`. Jede dieser Klassen hat eigene Attribute, wie `gruppe` und `abmessung` der Klasse `RahmenTeil`. Dass es sich bei `Komponente` um eine abstrakte Klasse handelt, bedeutet, dass man von ihr keine Objekte erzeugen kann. Aber die Objekte ihrer Subklassen sind ebenfalls vom Datentyp `Komponente`, was für den gemeinsamen Aufruf von Methoden und Attributen vorteilhaft ist. Die `toString()`-Methoden sind polymorphe Methoden, sie haben die gleiche Signatur. So wird erst zur Laufzeit der Software entschieden, welche davon für ein konkretes Objekt verwendet wird.	2	2	
	erläutern Die abgebildete Beziehung ist eine bidirektionale rekursive Assoziation auf das eigene Objekt. Jede Komponente kann eine vorherige und/oder eine nächste Komponente haben. In der Klasse `Komponente` werden `vorherige` und `naechste` als Referenzvariablen bereitgestellt. Die so verketteten Komponenten sind über ihre Vorgänger und über ihre Nachfolger erreichbar und können deshalb in beide Richtungen durchlaufen werden.		2	
1.2	überführen, implementieren			

```
public abstract class Komponente {
  private String iD;
  protected String bezeichnung;
  protected int gewicht;
  protected float preis;
  protected String hersteller;
  private Komponente naechste;
  private Komponente vorherige;

  public Komponente (String part0) {
  //part0 wird am Komma gesplittet und ergibt ein Array aus
  //5 Elementen für die 5 Attribute:
    String[] temp = part0.split(",");
    this.iD = temp[0];
    this.bezeichnung = temp[1];
    this.gewicht = Integer.parseInt(temp[2]);
    this.preis = Float.parseFloat(temp[3]);
    this.hersteller = temp[4];
  }

  public String toString() {
    return "ID=" + iD + ", " + bezeichnung + " wiegt " +
           gewicht + " g, kostet " + preis + " EUR,
           Hersteller: " + hersteller + ",";
  }
}
```

Aufg.	erwartete Leistungen	BE I	BE II	BE III

```
public class RahmenTeil extends Komponente {
  private String gruppe;
  private String abmessung;

  public RahmenTeil (String part0, String part1) {
    super(part0);
    String[] temp = part1.split(",");
    this.gruppe = temp[0];
    this.abmessung = temp[1];
  }

  public String toString() {
    return super.toString() + " " + gruppe + ", " +
           abmessung + " cm";
  }
}
```

Aufg.	erwartete Leistungen	I	II	III
	überführen	3		
	implementieren		3	2

1.3 entwickeln, zeichnen

b : Bike
bikeID = 104
anzahlKomp = 28
modell = Manu ATTAIN GTC Carbon

kat1 : Kategorie
bezeichnung = RacingBike
versicherung = false

ersteKomp

letzteKomp

rahmen1 : RahmenTeil
iD = r3
bezeichnung = Rahmen
gewicht = 1114
preis = 1000.50
hersteller = Manu
gruppe = Manu A
abmessung = 56 cm

vorherige

sch1 : SchaltElement
iD = s12
bezeichnung = Kurbel
gewicht = 641
preis = 222.69
hersteller = Champions
gruppe = Form 12
detail = 2 fach

naechste

rahmen2 : RahmenTeil
iD = r7
bezeichnung = Gabel
gewicht = 367
preis = 299.30
hersteller = Manu
gruppe = Manu A
abmessung = 1 1/8"-1 1/4"

naechste

vorherige

vorherige

einzel1 : EinzelTeil
iD = e13
bezeichnung = Sattel
gewicht = 187
preis = 50.00
hersteller = MFactory
name = BergWork
abmessung = Breite L

naechste

Aufg.	erwartete Leistungen	I	II	III
	entwickeln		2	2
	zeichnen	4		
1.4	implementieren		6	6

```
public class Bike {
  public static int autowert=0;
```

Aufg.	erwartete Leistungen	BE I	BE II	BE III
	(code below)			

```
  private int bikeID;
  private int anzahlKomp;
  private String modell;
  private String[] konfigzeilen;
  private Komponente ersteKomp;
  private Komponente letzteKomp;
  private Kategorie kat;

  public Bike(String dateiname) {
    this.leseDatenAusDatei(dateiname);
    //hier für das erste Bauteil:
    String[] teile = this.konfigzeilen[0].split(";");
    this.ersteKomp = new RahmenTeil(teile[0], teile[1]);
    this.letzteKomp = this.ersteKomp;
    //für die beiden Attribute in der letzten Konfigzeile:
    String[] rest =
      this.konfigzeilen[konfigzeilen.length- 1].split(";");
    this.modell = rest[0];
    kat = new Kategorie (rest[1]);
    this.anzahlKomp = 1;
    autowert ++ ;
    this.bikeID = autowert;
    while (this.hinzufuegenKomp()) {
    }
  }

  private void leseDatenAusDatei(String dateiname) {
    //nicht zu bearbeiten
  }

  public boolean hinzufuegenKomp() {
    //doppelte Verkettung, diese Methode fügt hinten an
    //erste Komponente ist bereits vorhanden (Konstruktor)
    Komponente komp = null;
    //falls noch nicht alle Komponenten montiert sind, wird
    //die nächste genommen:
    if (this.anzahlKomp >= this.konfigzeilen.length- 1 )
      return false;
    String[] teile =
              this.konfigzeilen[this.anzahlKomp].split(";");
    if (teile[0].startsWith("r"))
      komp = new RahmenTeil (teile[0],teile[1]);
      else if (teile[0].startsWith("s"))
        komp = new SchaltElement (teile[0], teile[1]);
      else if (teile[0].startsWith("e"))
        komp = new EinzelTeil (teile[0], teile[1]);
    //an letzteKomp anfügen:
    this.letzteKomp.setNaechste(komp);
    komp.setVorherige(this.letzteKomp);
    this.letzteKomp=komp;
    this.anzahlKomp ++ ;
    return true;
  }
}
```

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td>1.5</td><td>

implementieren

```java
public String ersetzeKomp(Komponente neu) {
    String ausbau = "Auszubauen sind:\n";
    if (letzteKomp.getID().equals(neu.getID())) {
      ausbau += letzteKomp.toString();
      neu.setVorherige(letzteKomp.getVorherige());
      letzteKomp.getVorherige().setNaechste(neu);
      letzteKomp.setVorherige(null);
      this.letzteKomp=neu;
    } else {
      Komponente aktuelle = letzteKomp;
      while (aktuelle != ersteKomp &&
            !aktuelle.getID().equals(neu.getID())) {
        ausbau += aktuelle.toString() ;
        aktuelle = aktuelle.getVorherige();
      }
      if(aktuelle != ersteKomp) {
        ausbau += aktuelle.toString();
        neu.setVorherige(aktuelle.getVorherige());
        neu.setNaechste(aktuelle.getNaechste());
        aktuelle.getVorherige().setNaechste(neu);
        aktuelle.getNaechste().setVorherige(neu);
        aktuelle.setVorherige(null);
        aktuelle.setNaechste(null);
      } else {
        ausbau = "Komponente nicht ausgebaut";
      }
    }
  return ausbau;
}
```

</td><td></td><td>3</td><td>4</td></tr>
<tr><td>1.6</td><td>entwickeln, zeichnen

siehe Anlage zu Aufgabe 1.6

entwickeln
zeichnen</td><td>4</td><td>2</td><td>3</td></tr>
<tr><td>1.7.1</td><td>beschreiben
Ein Anwendungsfalldiagramm modelliert die Anforderungen an ein System und dient der Absprache zwischen dem Kunden als Auftraggeber und dem Softwareentwickler als Auftragnehmer. Es beschreibt die Beziehungen zwischen Akteuren und Funktionalitäten in einem System. Auch Beziehungen zwischen Funktionalitäten können eingetragen werden.

einordnen
UML-Anwendungsfalldiagramme werden in der Analysephase des Softwarelebenszyklus eingesetzt. In der Integrationsphase wird getestet, ob das Endprodukt die festgelegten Anforderungen gewährleistet.</td><td>1</td><td>1</td><td></td></tr>
</table>

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
1.7.2	entwickeln, zeichnen BikeVerwaltung Bike montieren extension points Test-Bike <<include>> Bike konfigurieren Kunde <<include>> Bike ausgewählt? Bike ausliefern extension points Zubehör hinzufügen <<extend>> Komponente tauschen <<extend>> <<extend>> Servicepersonal Zubehör gewünscht? Komponente fehlerhaft? <<include>> Techniker Bike testen extension points Komponententausch Preis ermitteln Zubehör anbauen			
	entwickeln		2	2
	zeichnen	4		
	Summe 60	**18**	**23**	**19**

Aufg.	erwartete Leistungen	BE		
		I	II	III

2.1	angeben
	Möglichkeit 1:
	`Komponente(id, bezeichnung, gewicht, preis, hersteller, Gruppe, Abmessung, Detail, modellName)`
	
	Möglichkeit 2:
	`Rahmenteil(iD, bezeichnung, gewicht, preis, hersteller, gruppe, abmessung)`
	`Schaltelement(iD, bezeichnung, gewicht, preis, hersteller, gruppe, detail)`
	`Einzelteil(iD, bezeichnung, gewicht, preis, hersteller, modellName, abmessung)`
	
	Möglichkeit 3:
	`Komponente(iD, bezeichnung, gewicht, preis, hersteller)`
	`Rahmenteil(iD#, gruppe, abmessung)`
	`Schaltelement(iD#, gruppe, detail)`
	`Einzelteil(iD#, modellName, abmessung)`
		3		

	diskutieren Möglichkeit 1 würde zu einer einzigen Relation mit sehr vielen Attributen führen, wobei die Tabelle in jedem Fall viele null-Einträge hätte. Bei Möglichkeit 2 hat man einfache Zugriffe auf alle Attribute. Ein Nachteil wären gleiche IDs, zum Beispiel von Rahmenteilen und Einzelteilen, was hier aber ausgeschlossen ist. Bei Möglichkeit 3 wird jeder Entitätstyp in eine Tabelle überführt. Spezialisierungen besitzen dieselben Primärschlüssel wie die Generalisierung, diese sind zugleich auch Fremdschlüssel. Dieses Vorgehen ist einfach und systematisch, es hat nur den Nachteil, dass die speziellen Attribute der Bauteile durch umständliche Navigation erreichbar sind, dafür aber den Vorteil, die generellen Attribute (z.B. `preis`) über die dafür relevante Tabelle `Komponente` einfach erreichen zu können.			3

2.2	überführen
	`Fachhändler(fiD, name, plz, ort, strasseNr)`
	`Bike(bikeID, modell, baujahr, fiD#, kBez#)`
	`Kategorie(kBez, versicherung)`
	`Montage(miD, bikeID#, iD#)`
	`Komponente(iD, bezeichnung, gewicht, preis, hersteller)`
	`Rahmenteil(iD#, gruppe, abmessung)`
	`Schaltelement(iD#, gruppe, detail)`
	`Einzelteil(iD#, modellName, abmessung)`
		5		

2.3.1	formulieren
	`UPDATE Komponente SET preis = preis*1.05`
	`WHERE hersteller = 'MANU';`
			2	

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td>2.3.2</td><td>formulieren
<pre>
INSERT INTO Kategorie
VALUES('E-Bike Trekking',true);
INSERT INTO Fachhändler (name, plz, ort, strasseNr)
VALUES('MyChain', '63452', 'Hanau', 'Einkaufsstraße 28');
INSERT INTO Bike (modell, baujahr, fiD, kBez)
VALUES('Manu SportABC', YEAR(NOW()),
      (SELECT fiD
         FROM Fachhändler
        WHERE name = 'MyChain'
          AND ort = 'Hanau'),'E-Bike Trekking');
</pre></td><td></td><td>5</td><td>1</td></tr>
<tr><td>2.3.3</td><td>formulieren
<pre>
SELECT b.bikeID, b.modell, b.kBez
FROM Bike b, Fachhändler f
WHERE b.fiD = f.fiD
AND f.name = 'Bosse Zweirad'
AND f.ort = 'Köln'
AND b.baujahr = 2021;
</pre></td><td></td><td>3</td><td></td></tr>
<tr><td>2.3.4</td><td>implementieren
<pre>
SELECT b.bikeID, SUM(kom.gewicht) AS 'Gesamtgewicht'
FROM Bike b, Montage m, Komponente kom
WHERE b.bikeID = m.bikeID AND m.iD = komp.iD
      AND b.kBez = 'RacingBike'
GROUP BY b.bikeID;
</pre></td><td></td><td>2</td><td>1</td></tr>
<tr><td>2.3.5</td><td>entwickeln
<pre>
SELECT k.hersteller, COUNT(k.iD) AS Anzahl
FROM Bike b, Montage m, Komponente k, Schaltelement s
WHERE b.bikeID = m.bikeID AND m.iD = k.iD AND k.iD = s.iD
      AND s.detail LIKE 'Felgen-Br'
      AND b.baujahr = YEAR(NOW())-1
GROUP BY k.hersteller
HAVING COUNT(k.iD)>=10
ORDER BY Anzahl DESC;
</pre></td><td></td><td>2</td><td>3</td></tr>
<tr><td></td><td>Summe 30</td><td>8</td><td>14</td><td>8</td></tr>
</table>

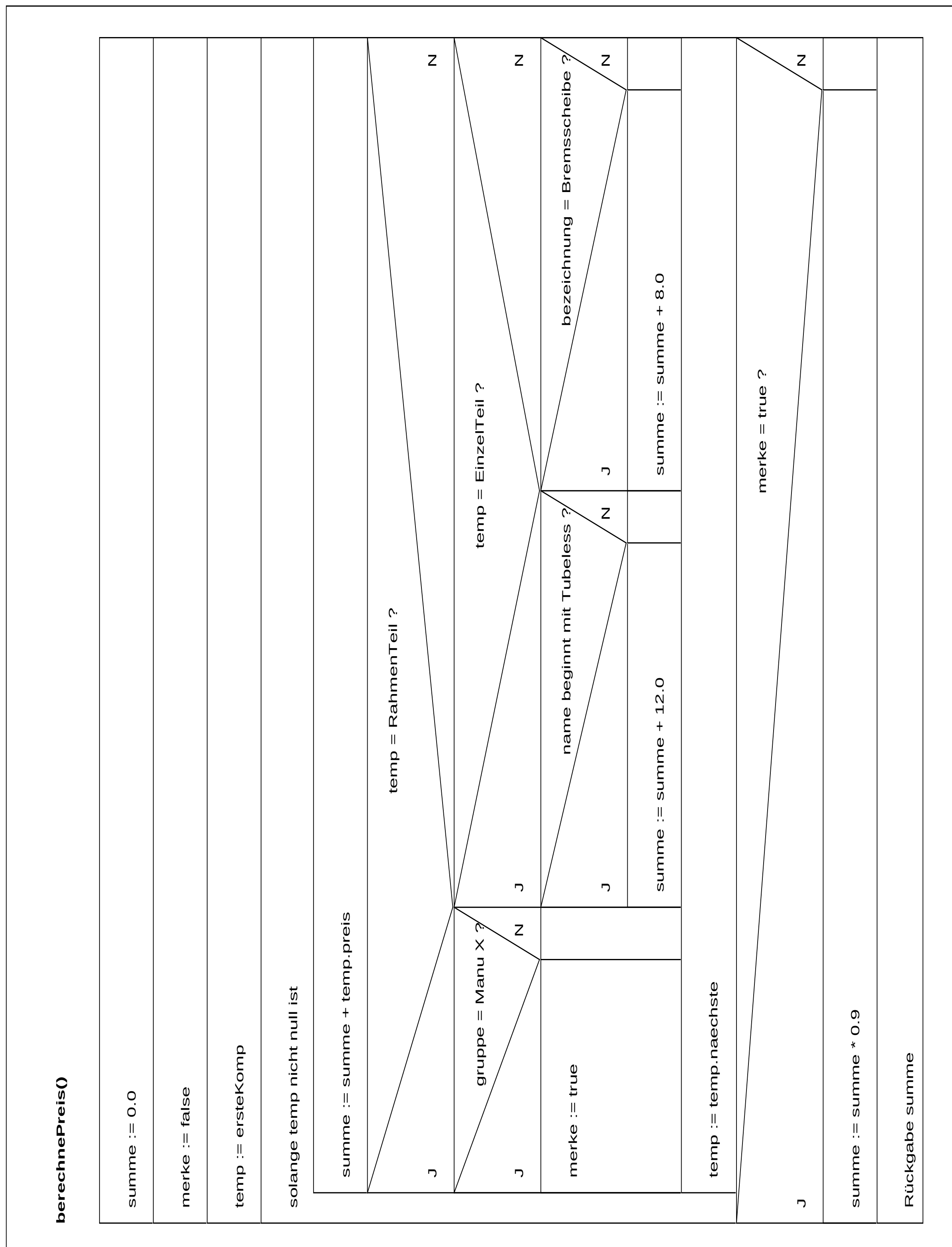
Nicht für den Prüfling bestimmt

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE I	BE II	BE III
3	entwickeln, zeichnen Entitäten: Hersteller (hiD, ort, name); Zubehör (ziD, bezeichnung, marke, preis); Bike (bikeID, zustand, rahmengröße, verfügbarkeit, laufradgröße, farbe, gruppe); Leasingvertrag (vNr, monatlVers, leasingrate, beginn, ende); Kunde (kiD, name, adresse); Arbeitgeber (aiD, firmenname) Beziehungen: Hersteller [1,1] – liefert (ekPreis, datumLief) – [1,n] Bike; Bike [0,n] – wird ergänzt – [0,m] Zubehör; Bike [5,n] – betrifft – [0,1] Leasingvertrag; Bike [0,n] – wird verkauft (datumVerk, vPreis) – [0,1] Kunde; Leasingvertrag [0,n] – schließt ab – [1,1] Arbeitgeber			
	entwickeln		3	3
	zeichnen	3	1	
	Summe 10	**3**	**4**	**3**

Nicht für den Prüfling bestimmt

Anlage zu Aufgabe 1.6



III Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt unter Beachtung der nachfolgenden Vorgaben nach § 33 der Oberstufen- und Abiturverordnung (OAVO) in der jeweils geltenden Fassung. Bei der Bewertung und Beurteilung der sprachlichen Richtigkeit in der deutschen Sprache sind die Bestimmungen des § 9 Abs. 12 Satz 3 OAVO in Verbindung mit Anlage 9b anzuwenden.

Bei der Bewertung und Beurteilung der Übersetzungsleistung in den Fächern Latein und Altgriechisch sind die Bestimmungen des § 9 Abs. 14 OAVO in Verbindung mit Anlage 9c anzuwenden.

Der Fehlerindex ist nach Anlage 9b zu § 9 Abs. 12 OAVO zu berechnen. Für die Ermittlung der Punkte nach Anlage 9a zu § 9 Abs. 12 OAVO sowie Anlage 9c zu § 9 Abs. 14 OAVO wird jeweils der ganzzahlige nicht gerundete Prozentsatz bzw. Fehlerindex zugrunde gelegt.

Für die Bewertung in den modernen Fremdsprachen ist der „Erlass zur Bewertung und Beurteilung von schriftlichen Arbeiten in allen Grund- und Leistungskursen der neu beginnenden und fortgeführten modernen Fremdsprachen in der gymnasialen Oberstufe, dem beruflichen Gymnasium, dem Abendgymnasium und dem Hessenkolleg“ vom 7. August 2020 (ABl. S. 519) zugrunde zu legen. Demnach erfolgt die Bewertung und Beurteilung mit der Maßgabe, dass lediglich bei der Ermittlung des Prüfungsergebnisses (Note) aus Prüfungsteil 1 und 2 gerundet wird.

Darüber hinaus sind die Vorgaben der Erlasse „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen (Abiturerlass)“ und „Durchführungsbestimmungen zum Landesabitur“ in der für den Abiturjahrgang geltenden Fassung zu beachten.

Als Kriterien für die Bewertung und Beurteilung dienen unter Beachtung der Zielsetzung der gymnasialen Oberstufe nach § 1 Abs. 2 OAVO neben dem Inhaltlichen auch die in den Kerncurricula genannten überfachlichen Kompetenzen, insbesondere die Sprachkompetenz und Wissenschaftspropädeutik; dies zeigt sich u.a. in qualitativen Merkmalen wie Strukturierung, Differenziertheit, (fach-)sprachlicher Gestaltung und Schlüssigkeit der Argumentation.

Im Fach Praktische Informatik besteht die Prüfungsleistung aus der Bearbeitung eines Vorschlags, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass mindestens 45% der zu vergebenden BE erreicht werden. Ein Prüfungsergebnis von **11 Punkten (gut)** setzt voraus, dass mindestens 75% der zu vergebenden BE erreicht werden.

Gewichtung der Aufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen

Aufgabe	**Bewertungseinheiten in den Anforderungsbereichen**			**Summe**
	AFB I	**AFB II**	**AFB III**	
1	18	23	19	**60**
2	8	14	8	**30**
3	3	4	3	**10**
Summe	**29**	**41**	**30**	**100**

Die auf die Anforderungsbereiche verteilten Bewertungseinheiten innerhalb der Aufgaben sind als Richtwerte zu verstehen.